Vol.2018-HPC-167 No.8
2018/12/17

Gooseコンパイラによる PEZY-SC2 クラスタ睡蓮 2 での応用計算

中里 直人[1]　台坂 博[2]　湯浅 富久子[3]　石川 正[3]

概要：本発表では, ディレクティブ型コンパイラ Goose による PEZY-SC2 クラスタ睡蓮 2 での応用計算ついて報告する. Goose は C++で記述されたソースコード内のディレクティブで指定された二重ループを GPU, PEZY-SC2, FPGA などのアクセラレータで並列計算するコンパイラである. Goose により素粒子物理学におけるファインマンループ積分の数値計算が様々なアクセラレータで容易に可能となった. 具体的な応用の計算方法として, 二重指数積分法と準モンテカルロ法による多次元の数値積分を睡蓮 2 で実行したときの性能評価と比較について報告する.

キーワード：高精度計算, OpenCL, アクセラレータ

1. はじめに

Goose は C/C++で記述されたプログラムから, 指定されたループを並列化するディレクティブ型のコンパイラである. メニーコアアクセラレータの GRAPE-DR[6], [7] をターゲットとして開発され, その拡張として多倍長精度専用計算機 GRAPE-MP[2] や GRAPE9-MPX[18], [19] への対応をした. さらにループ部分に対応する OpenCL カーネルを生成するように拡張することで, OpenCL 対応デバイス (GPU やマルチコア CPU など) でも利用できるようになった.

Goose は二重ループによる総和計算に特化したコンパイラである. 天文学/天体物理学での重力多体問題や分子動力学では, 互いに相互作用する粒子の軌道進化を計算するため, 運動方程式を数値積分する. このような場合における, 粒子間の相互作用による力の総和計算が Goose により並列化できるループの典型例である. Goose を使うことで, 様々なアクセラレータを一つのソースコードから利用することができる. 二重ループによる総和計算は, 多体問題だけではなく, 本論文で紹介する多重積分の数値計算や, モンテカル法による数値積分にも適用可能である.

図 1 は, Goose でのコンパイル処理のフローを示す. フロントエンドは, C/C++のソースコードを構文解析して, ディレクティブが指定されたループの再内側の演算を切り出し, 指定されたターゲット用の命令コード (GRAPE-DR および GRAPE9-MPX の場合) または OpenCL カーネルを生成する. 元のソースコードの指定されループの部分は, アクセラレータに処理をオフロードするための API 呼び出しに置き換えられる. GRAPE-DR および GRAPE9-MPX の場合には図にある LSUMP[9] により, コードの最適化をした上で命令コードを生成する. OpenCL のカーネルは, ターゲットとして PEZY-SC/PEZY-SC2[16] に対応する PZCL カーネルを生成することもできる. 以上のいずれの場合にも, 複数のアクセラレータボードに計算を分割して並列計算するために, ホスト側コードを MPI による並列化することも可能である. GRAPE9-MPX は 8 台の FPGA を搭載したホスト計算機 8 台からなるシステムであり [19], プロセスごとに 1 台の FGPA を割り当て, 64 プロセスの MPI 並列化を実現した. 同様に, GPU クラスタ上でも MPI 並列化を組み合わせたプログラムを実行可能である. 台坂ら [1] は Goose コンパイラを利用し GRAPE9-MPX により四倍精度演算器により応用計算をおこなった.

本報告では, PEZY-SC2 を採用した並列コンピュータシステムである睡蓮 2 での, Goose による応用計算の性能評価について報告する. 応用例として素粒子反応のファインマンループ積分の数値計算についての結果を示す.

2. ファインマンループ積分の数値積分

ファインマン図は, 素粒子反応の反応過程をあらわす図である (図 2,3). この図は, 頂点とそれらを接続する外線, 内線から構成されており, 外線は素粒子反応の始状態と終

[1] 会津大学
[2] 一橋大学
[3] 高エネルギー加速器研究機構

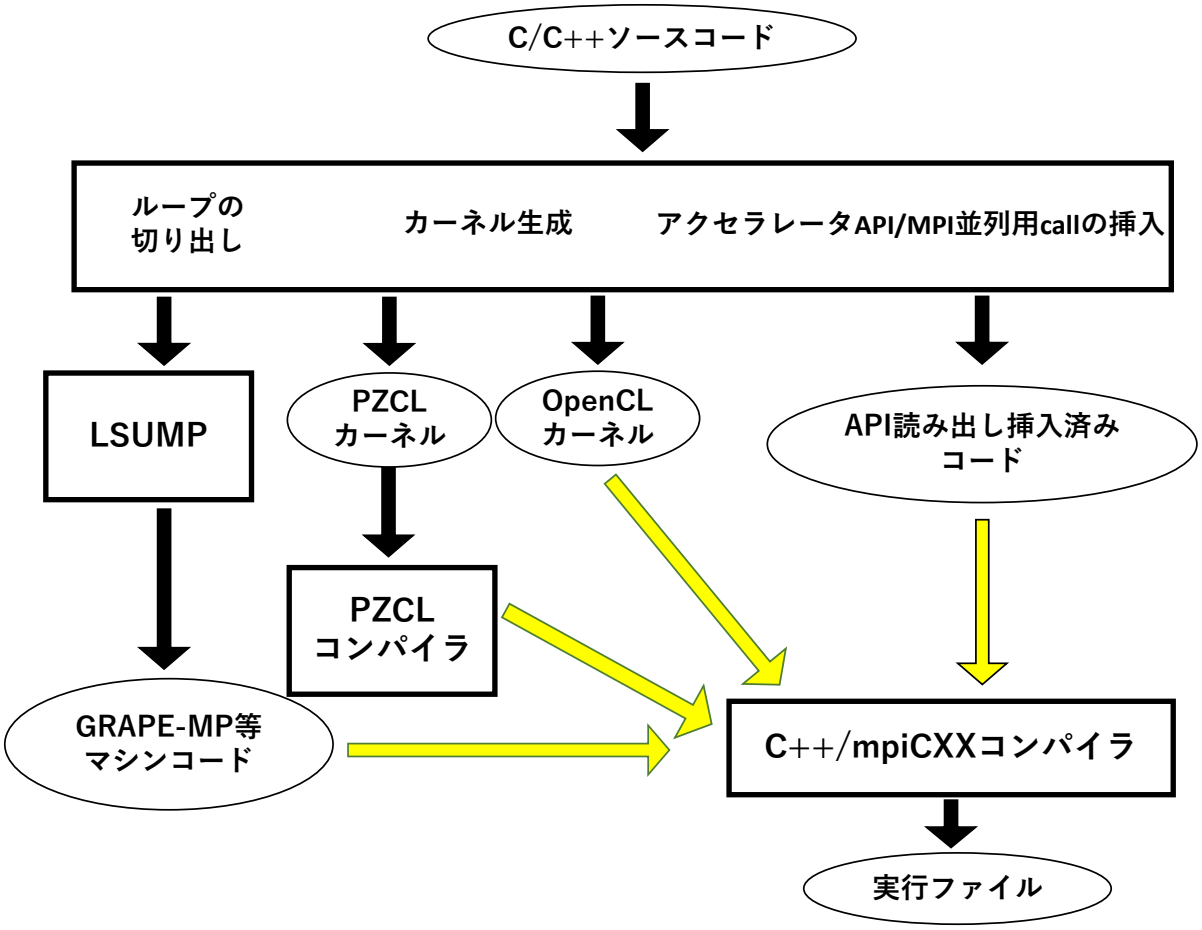

図 1　Goose コンパイラによるソースコード変換と実行ファイル生成の流れ

状態を表している. 図 2,3 で, 両側にある水平線が外線である. 内線は素粒子の伝搬をあらわし, その接続の仕方により図には閉じたループができる. 二つの素粒子間の反応過程には様々な場合があり, その場合ごとにファインマン図が対応する. 反応過程の摂動の高次の補正を求めるためには, ループを含むファインマン図について積分 (ファインマンループ積分; 以下, ファインマン積分) を求める必要がある.

L 個のループと N 個の内線のあるファインマン図に対応するファインマン積分 I は以下の式で表される.

$$I = G \int_0^1 \prod_{l=1}^{N} dx_l \delta(1 - \sum x_i) \frac{C^{N-d(L+1)/2}}{(D - i\rho C)^{N-dL/2}}, \quad (1)$$

ここで G は以下の式で定義される.

$$G = \frac{\Gamma(N - \frac{dL}{2})}{(4\pi)^{\frac{dL}{2}}} (-1)^N, \quad (2)$$

式 (1) の被積分関数の分母の i は複素数単位を表す. また, $\delta()$ はデルタ関数, $\Gamma()$ はガンマ関数を表す. C と D はファインマン変数 x_l の多項式であり, ファインマン図の形状ごとに決まる. 積分 I はデルタ関数の処理をすると $N-1$ 重積分に帰着する. $d = 4$ は時空の次元であり, ρ は被積分関数の分母がゼロになることによる発散を正則化するパラメータである.

本研究では多様にあるファインマン積分を計算することが最終的な目標である. 被積分関数の分母がゼロになることがない場合は $\rho = 0$ として数値積分を計算する. そうではない場合, ファインマン積分を ρ の関数 $I(\rho)$ として, $\rho \to 0$ の場合の漸近値を数値計算する必要がある. そのためには, ρ を変えて $I(\rho)$ の数列を数値積分により求め, 補外法を適用する [14]. この場合, ひとつのファインマン積分について, 例えば $\rho_\beta = 1.15^{-\beta}$ として, β を徐々に大きくして 50 回程度変化させて $I(\rho_\beta)$ の数列を得る必要がある.

いずれの場合にも $I(\rho_\beta)$ を数値計算で求める方法として有効な手法は, (a) 適応型積分法 [4], (b) 二重指数積分法 [15], [17] (c) 準モンテカルロ計算法 [3] がある. このうち (b) と (c) は, どちらも多重積分を直接的に計算する方法であり, Goose コンパイラによる並列化が可能なループをもつ総和計算となる. 以下, $\boldsymbol{x} = (x_1, x_2, x_3..)$ とし, 式 (1) の被積分関数を $f(\rho_\beta, \boldsymbol{x})$ とする.

2.1　二重指数積分法の詳細

二重指数 (Double Exponetial, 以下 DE) 積分法 [8] は, DE 変換 (例えば $x = \phi(t) = \tanh(\frac{\pi}{2}\sinh(t))$) と台形公式を組み

合わせる数値積分法である. 台形公式のきざみ幅を h とし $N=3$ の場合, ファインマン積分 $I(\rho_\beta)=G\int f(\rho_\beta,\boldsymbol{x})d\boldsymbol{x}$ の数値積分は以下の二重の総和計算となる.

$$I_{\mathrm{DE}}(\rho_\beta)=h^2\sum_{j=-\infty}^{\infty}\phi'_j\sum_{i=-\infty}^{\infty}f(\rho_\beta,\phi_i,\phi_j)\phi'_i \qquad (3)$$

ここで $\phi_i=\phi(ih)$ 等であり, 変換関数 $\phi(t)$ の微分 $\phi'(t)$ についても $\phi'_i=\phi'(ih)$ 等とする. 通常 i,j は有限の項数の和をとり, 例えば $-n/2\le i,j\le n/2$ とすると, この数値積分を計算するためには被積分関数と ϕ'_i の積を $O(n^2)$ 回計算する必要がある. 変数変換した空間での積分点 (ϕ_i,ϕ_j) と ϕ'_i 等の数列は, 各空間軸で等間隔かつ同一であり事前に計算しておくことができる. アクセラレータで計算する際には, この数列をホスト計算機で計算してアクセラレータのメモリに配置することで, 計算コストの高い指数関数を繰り返し計算する必要はなくなる. さらに, 異なる j ごとに再内側のループを個々のスレッド (カーネル) で計算することで非常に効率よく計算できる. $N>3$ の場合でも, 総和を融合することで形式的に二重の総和に変換できるため, Goose コンパイラにより並列計算できる.

2.2 準モンテカルロ計算法

モンテカルロ (Monte Calro; MC) 法による数値積分では, ファインマン積分 $I(\rho_\beta)$ の近似値をランダムに生成した積分点 $\boldsymbol{x}_j, j=1,2,3,...P$ について

$$I_{\mathrm{MC}}(\rho_\beta)=\frac{1}{P}\sum_{j=1}^{P}f(\rho_\beta,\boldsymbol{x}_j) \qquad (4)$$

という総和計算として計算する. 単純に乱数により積分範囲 $0\le x\le 1$ で積分点を生成する場合, I_{MC} の積分誤差は $O(P^{-\frac{1}{2}})$ となり, P を大きくしたときでも収束が遅いという欠点がある.

準モンテカルロ (Quasi Monte Calro; QMC) 計算法では, 積分空間内を乱数より効率よくカバーするようなルールに基づいて積分点を生成する. そのひとつである rank-1 lattice ルールの場合は

$$\boldsymbol{x}_j=\{\frac{j}{P}\boldsymbol{z}\}=\frac{j\boldsymbol{z}\bmod P}{P} \qquad (5)$$

により積分点を計算する [13]. ここで $\boldsymbol{z}$ は整数のベクトルであり, 積分点を計算するための生成ベクトルと呼ばれる. 中括弧 {} はベクトルの各成分の小数点以下の値を得ることを意味し, 結局二つ目の等式が得られる. $\boldsymbol{z}$ を計算する手法として Fast CBC(component-by-component) アルゴリズム [10], [11] があり, 必要な P に対して事前に計算しておく必要がある. 例えば $N=5$ で $P=10M$ 点の時には $\boldsymbol{z}=(1,2928962,1859617,3250721)$ である [3]. 準モンテカルロ計算法での積分誤差は $O(P^{-1})$ となり, より収束が速い.

QMC 法でファインマン積分を計算する場合には, Sidi 変換 [12] 等により変数変換をし, 積分範囲端点での被積分関数の特異性を取り除く. de Doncker[3] によると, 例えば 6 次の Sidi 変換

$$\Phi(t)=t-(45\sin(2\pi t)-9\sin(4\pi t)+\sin(6\pi t))/(60\pi) \quad (6)$$

を利用する. この時 $N=3$ のファインマン積分は

$$I_{\mathrm{QMC}}(\rho_\beta)=\frac{1}{P}\sum_{j=1}^{P}f(\rho_\beta,\Phi_1^j,\Phi_2^j)\Phi'^j_1\Phi'^j_2 \qquad (7)$$

となる. ここで $\boldsymbol{z}=(z_1,z_2)$ として, $x_1^j=\{\frac{j}{P}z_1\}$ 等とする. また $\Phi_1^j=\Phi(x_1^j)$ およびその微分 $\Phi'^j_1=\Phi'(x_1^j)$ 等とする. 結果, 被積分関数と Φ' の積を P 回計算する必要がある. Sidi 変換を使った QMC 法では, DE 法とは異なり積分点も変換の微分関数も事前に計算しておくことはできない. アクセラレータで計算する際には, Fast CBC アルゴリズムで求めた $\boldsymbol{z}$ を与えて, 各スレッドではスレッド番号から積分点 (x_1^j,x_2^j) を計算する.

アクセラレータでの演算効率を上げるためには, スレッドごとの演算量を多くするよう総和を以下の二重の総和に書き換える.

$$I_{\mathrm{QMC}}(\rho_\beta)=\frac{1}{P}\sum_{k=0}^{Q-1}\sum_{j=k\Delta Q+1}^{(k+1)\Delta Q}f(\rho_\beta,\Phi_1^j,\Phi_2^j)\Phi'^j_1\Phi'^j_2 \qquad (8)$$

ここで Q を総スレッド数とし, $\Delta Q=P/Q$ とする. この場合, アクセラレータの各スレッドは ΔQ 点の部分和を計算する. 全体の総和は可能ならアクセラレータ, そうでない場合はホスト計算機でリダクション計算をして得る. 式 (8) の総和計算は Goose コンパイラにより並列計算できる. Goose コンパイラによる MPI 並列化は式 (8) と同様の手法により, 外側の総和をさらに分割するものである. そのため P が十分大きいならば、効率を落とすことなく並列化が可能である.

3. PEZY-SC2 クラスタ睡蓮 2 でのファインマン積分計算の性能評価

図 2 と図 3 に今回の研究で対象としたファインマン図を示す. いずれの場合も内線が 8 本あり, ループは 3 カ所ある. よって $N=8$, $L=3$, $d=4$ となり, ファインマン積分は

$$I(\rho)=G\int_0^1\frac{C}{(D-i\rho C)^2}d\boldsymbol{x} \qquad (9)$$

という形式の多重積分である. C 式は $\boldsymbol{x}$ のみによる L 次の多項式, D 式は $\boldsymbol{x}$ とパラメータ $\boldsymbol{m}=(m_1,m_2,...m_8)$ と s を含む $L+1$ 次の多項式となる. 質量パラメータ $\boldsymbol{m}$ は個々のファインマン変数に対応したパラメータである. 運動学変数パラメータ s は素粒子反応の条件で決まるパラメータ

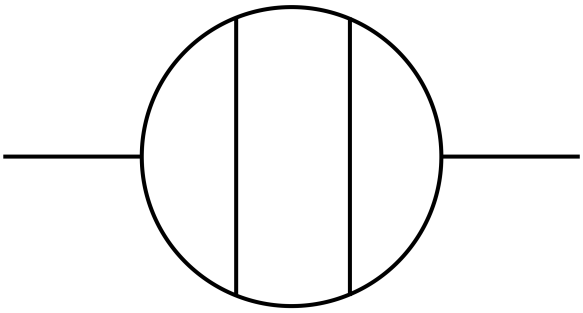

図 2　DE 法の性能評価のためのファインマン図

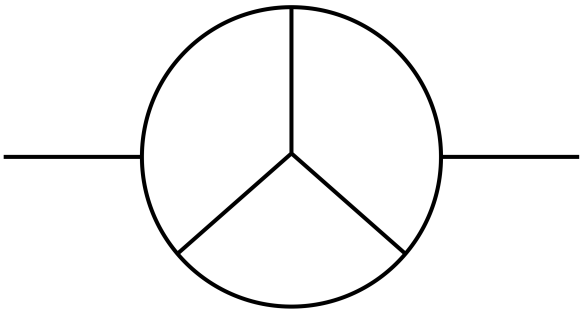

図 3　QMC 法の性能評価のためのファインマン図

である. $\boldsymbol{m}$ や s の値により, D 式が積分区間内でゼロとならない場合には $\rho = 0$ としてよい.

3.1　PEZY-SC クラスタ睡蓮 2

睡蓮 2 は高エネルギー加速器研究機構に設置されている PEZY-SC2 プロッサ採用したスーパーコンピュータシステムである. PEZY-SC2 プロセッサは, PEZY Computing 社により開発された Multiple Instruction Multiple Data(MIMD) 方式のプロセッサであり, ホスト計算機と組み合わせて演算アクセラレータとして利用する. 睡蓮 2(機種名 ZettaScaler-2.2) の計算ノードは Intel Xeon D-1571 を 2 個と PEZY-SC2 プロセッサが 8 台から構成されている. PEZY-SC2 プロセッサあたり 32GB の DDR4 メモリを搭載する. これらのノード間は InfiniBand FDR で接続されている. ノードは液浸冷却用筐体に収められており, 液体を循環させることでシステム全体を直接冷却する. 2018 年 6 月に発表された TOP500 では, Rmax 性能 797.994 TFLOPS(Rpeak 性能 1082.57 TFLOPS) で 419 位であった. 同時期の GREEN500 ランキングでは 16.835 GFLOPS/W で 2 位であった. PEZY-SC2 及び ZettaScaler 計算機についての詳細は鳥居らの論文を参照 [16].

3.2　DE 法の性能評価

DE 法の性能評価には図 2 に対応するファインマン積分を使った. このファインマン図では $\rho \neq 0$ であり, ファインマン積分を DE 法で計算する際には実数部と複素数部を同時に計算する. また, 異なる ρ_β について多項式 C, D は同じであるため, 複数の β の計算を同時におこなうことで実効的な演算数を削減することができる. 図 4 は, 睡蓮 2 にて PEZY-SC2 プロセッサを 1 台使った場合の $n = 49, 64$ での経過時間を示す. 横軸は同時に計算した β のパターン数 (S) を示す. ひとつの β あたり実数部と複素数部の和を得るため, 2 の倍数ごとに時間計測をした. S が大きいほど計算時間は長くなるが, その傾きは小さい.

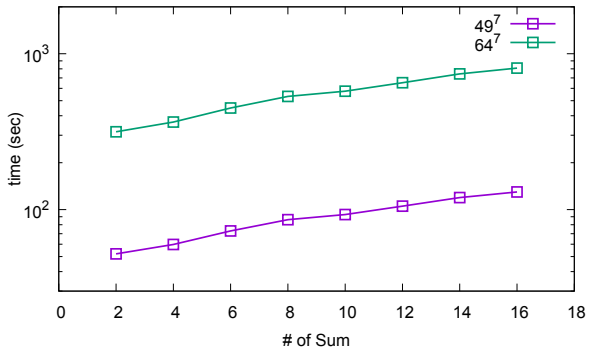

図 4　睡蓮 2 での DE 法による $n = 49, 64$ の場合の経過時間 (秒)

S	演算数	49^7	64^7
2	113	3.83×10^{-11}	3.59×10^{-11}
4	146	2.20×10^{-11}	2.07×10^{-11}
8	172	1.59×10^{-11}	1.51×10^{-11}
16	224	1.20×10^{-11}	1.15×10^{-11}

表 1　睡蓮 2 での DE 法によるカーネルの演算数と積分点あたりの経過時間 (秒)

表 3.2 にカーネルの演算数と, 積分点の総数は $2 \times S \times n^7$ より計算した積分点あたりの経過時間を示す. 積分点数は $n = 49, 64$ の時にそれぞれ $S \times 1.37 \times 10^{12}, S \times 8.79 \times 10^{12}$ 点となる. 式 C, D の計算を繰り返す必要がないため, S が大きいほど計算時間は短くなる. $S = 2$ の場合と比べて $S = 16$ の時の演算数はほぼ 2 倍に増加している. $S = 16$ と同等の計算を異なる β ごとに別のカーネルで計算した場合, 実効的には $113 \times 8 = 904$ 演算必要になる. よってこの手法により演算効率は約 4 倍向上している. またカーネルあたりの演算数が増えて演算密度が向上しアクセラレータには向いた演算となる. ただし S が大き過ぎる場合カーネル実行に必要な変数が増えるため, プロセッサのレジスタ及びローカルメモリに収まるよう S を選ぶ必要がある.

Goose コンパイラでは, アクセラレータで実行する際の演算精度を倍精度と倍々精度 (double-double 法 [5]) で切り替えることができる. $n = 64, S = 4$ の場合に, 倍精度および倍々精度計算の経過時間を表 3.2 に示す. 比較対象として Goose コンパイラで OpenCL カーネルを生成し, Nvidia 社の Tesla GPU P100 で同じ計算を実行した時の時間も示す. 睡蓮 2 の PEZY-SC2 プロセッサは GPU P100 と比べると約 2.3 倍の時間がかかっている. PEZY-SC2 プロセッサの

	倍精度	倍々精度
PEZY-SC2	532.5	8940.8
Tesla P100	228.5	3313.6

表 2 PEZY-SC2 と Tesla P100 での DE 法の計算時間の比較 (秒)

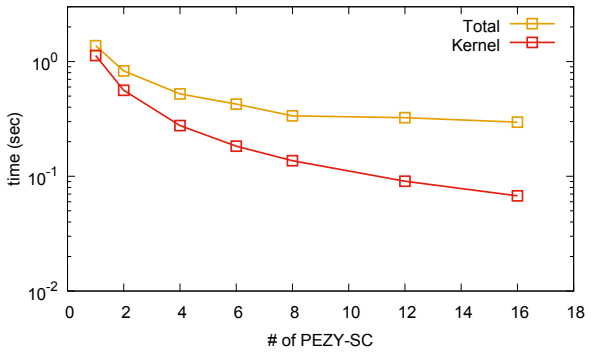

図 5 睡蓮 2 での QMC 法による MPI 並列計算の全経過時間 (Tolta) とカーネル (Kernel) 実行時間 (秒)

場合, 倍々精度による計算時間は倍精度と比べて約 16.8 倍の時間がかかっている. GPU では約 14.5 倍である. この差の主因は, PEZY-SC2 プロセッサは倍々精度乗算の実装に有効な Fused-Multiply-Add(FMA) 演算器を搭載していないことである.

3.3 QMC 法の性能評価

QMC 法の性能評価には図 3 に対応するファインマン積分を使った. このファインマン積分の計算では $\rho = 0$ となる質量パラメータを使用した. Goose により生成されたカーネルから演算数をカウントすると, 積分点の座標生成を含めて積分点あたり 273 演算であった. このカーネルを式 (8) の方法で部分和にわけて計算した. この計算では計算精度は倍精度とした.

$P \sim 4 \times 10^8$ 点の場合のプロセス数を変化させた時の計算時間を図 5 に示す. プロセスごとに 1 台の PEZY-SC2 プロセッサを割り当てた. 総和のリダクション計算はホスト側で MPI によりおこなっている. よって, 全経過時間はパラメータ初期化と, PEZY-SC2 とのデータのやりとり, さらに MPI によるリダクション計算を含んだ時間である. カーネル時間は QMC の計算カーネルだけの計算時間である. カーネル時間はプロセス数 (PEZY-SC2 の利用台数) に反比例して, 部分和の項数が減少するためよくスケーリングする. 他のオーバヘッドがあるため全経過時間のスケーリングはあまりよくない. この主因はアクセラレータの性能に対して積分点数が少ないためで, DE 法の場合と同じ程度の $O(10^9)$ の P で計算できるように現在検討している.

3.4 DE 法と QMC 法の比較

DE 法は多次元積分のどの変数についても同一の積分点の分布のため, 積分点での変換関数の微分値も事前にホスト計算機で計算してしておくことで, DE 変換のような計算コストの高い計算を省略可能である. 一方でそのデータをメモリ上に保持するためには, ホスト側とアクセラレータ側の両方に n のべき乗に比例する大量のメモリが必要である. よって, 高次の摂動計算に必要な N, L が大きいファインマン図に対応する積分を計算する歳には, メモリ容量の点で不利である. また, DE 変換により積分範囲の端点での特異性を解消することができるが, どのファインマン変数についても同一の分布を使うために, 積分区間内部に被積分関数の特異点 (被積分関数の分母がゼロの点) が存在する場合には不利である. 多項式 C, D はファインマン図のループ数 $L, L+1$ に比例した高次多項式となり, L が大きい場合は, 多変数多項式の根を効率よく求める手法は確立されていないため, 個々の特異点で積分区間を分割する手法は実用的ではない.

QMC 法は積分点自体をカーネル内部で計算するため, 大量のメモリは必要としてないため, N, L が大きい場合には有利である. 一方で, 必要な積分点数 P に対して生成ベクトルを事前に計算しておく必要があり, 積分点数を任意に大きくすることは容易にできない. また, そのために被積分関数の評価以外にも演算が必要という欠点がある. ふたつの手法はそれぞれ利点と欠点があり, 今後より高次の摂動を計算するためには, 両者を相補的に利用してさらなる高速化のための計算手法と並列化の研究および, 新規プロセッサ対応のための研究が必要である.

4. まとめ

本研究では, ディレクティブ型コンパイラ Goose による PEZY-SC2 クラスタ睡蓮 2 でのファインマンループ積分の数値計算の性能評価について報告した. ファインマンループ積分対する二重指数積分法と準モンテカルロ法によるアクセラレータでの計算高速化は, 最近ようやく実用的になったものであり, 同一の被積分関数について, 同等の計算条件での演算誤差の評価は今後の課題である. また, 積分の条件がより困難な高次の摂動項の評価に必要なファインマンループ積分を計算するためには, 倍々精度演算や多倍長精度浮動小数点演算での計算が必要となる.

謝辞

Goose コンパイラの実装および拡張は, 株式会社 K&F Computing Research の福重俊之博士および川井敦博士との協力および共同研究としておこなっている. また, ウェスタンミシガン大学の de Doncker 教授には準モンテカルロコードの性能評価で協力を得た. 本研究の一部は, 文部科学省「次世代領域研究開発」(高性能汎用計算機高度利用事業費補助金)「ヘテロジニアス・メニーコア計算機による大規模計算科学」の補助を受けている.

参考文献

[1] Daisaka, H., Nakasato, N., Ishikawa, T., Yuasa, F. and Nitadori, K.: A development of an accelerator board dedicated for multi-precision arithmetic operations and its application to Feynman loop integrals II, *Journal of Physics: Conference Series*, Vol. 1085, No. 5, p. 052004 (online), available from ⟨http://stacks.iop.org/1742-6596/1085/i=5/a=052004⟩ (2018).

[2] Daisaka, H., Nakasato, N., Makino, J., Yuasa, F. and Ishikawa, T.: *GRAPE-MP: An SIMD Accelerator Board for Multi-precision Arithmetic*, *Procedia Computer Science*, Vol. 4, pp. 878–887 (2011).

[3] de Doncker, E., Almulihi, A. and Yuasa, F.: High-speed evaluation of loop integrals using lattice rules, *Journal of Physics: Conference Series*, Vol. 1085, No. 5, p. 052005 (online), available from ⟨http://stacks.iop.org/1742-6596/1085/i=5/a=052005⟩ (2018).

[4] de Doncker, E., Fujimoto, J., Hamaguchi, N., Ishikawa, T., Kurihara, Y., Shimizu, Y. and Yuasa, F.: Quadpack computation of Feynman loop integrals, *Journal of Computational Science*, Vol. 3, No. 3, pp. 102 – 112 (online), DOI: https://doi.org/10.1016/j.jocs.2011.06.003 (2012). Scientific Computation Methods and Applications.

[5] Hida, Y., Li, X. and Bailey, D.: Algorithm for quad-double precision floating point arithmetic, *Proc. 15th IEEE Symposium on Computer Architecture*, pp. 287–302 (2001).

[6] Makino, J., Hiraki, K. and Inaba, M.: GRAPE-DR: 2-Pflops massively-parallel computer with 512-core, 512-Gflops processor chips for scientific computing, *SC07* (2007).

[7] Makino, J., Daisaka, H., Fukushige, T., Sugawara, Y., Inaba, M. and Hiraki, K.: The performance of GRAPE-DR for dense matrix operations, *Procedia Computer Science*, Vol. 4, pp. 888 – 897 (online), DOI: http://dx.doi.org/10.1016/j.procs.2011.04.094 (2011). Proceedings of the International Conference on Computational Science, ICCS 2011.

[8] Mori, M.: *Discovery of the Double Exponentioal Transformation and Its Developments*, *Publications of the Research Institute for Mathematical Sciences*, Vol. 41(4), pp. 897–935 (2005).

[9] Nakasato, N. and Makino, J.: *A Compiler for High Performance Computing With Many-Core Accelerators*, *IEEE International Conference on Cluster Computing and Workshops*, pp. 1–9 (2009).

[10] Nuyens, D. and Cools, R.: Fast Algorithms for Component-by-Component Construction of Rank-1 Lattice Rules in Shift-Invariant Reproducing Kernel Hilbert Spaces, *Mathematics of Computation*, Vol. 75, No. 254, pp. 903–920 (online), available from ⟨http://www.jstor.org/stable/4100318⟩ (2006).

[11] Nuyens, D. and Cools, R.: Fast Component-by-component Construction of Rank-1 Lattice Rules with a Non-prime Number of Points, *J. Complex.*, Vol. 22, No. 1, pp. 4–28 (online), DOI: 10.1016/j.jco.2005.07.002 (2006).

[12] Sidi, A.: Extension of a class of periodizing variable transformations for numerical Integration, *Mathematics on Computation*, pp. 327–343 (2006).

[13] Sloan, I. and Joe, S.: *Lattice Methods for Multiple Integration*, Oxford University Press (1994).

[14] Wynn, P.: *On the convergence and stability of the epsilon algorithm*, *SIAM Journal of Mathematical Physics*, Vol. 3, pp. 91–122 (1966).

[15] Yuasa, F., Ishikawa, T., Hamaguchi, N., Koike, T. and Nakasato, N.: *Acceleration of Feynman loop integrals in high-energy physics on many core GPUs*, *Journal of Physics: Conference Series*, Vol. 454, No. 1, IOP Publishing, p. 012081 (online), DOI: 10.1088/1742-6596/454/1/012081 (2013).

[16] 鳥居　淳，石川　仁，木村耕行，齊藤元章：グリーンスーパーコンピュータ ZettaScaler の技術と今後の展望，電子情報通信学会論文誌 C，Vol. J100-C, No. 11, pp. 537–544 (2017).

[17] 湯浅富久子，濱口信行：二重指数関数型積分法の素粒子物理学への応用，情報処理学会研究報告 (2008-HPC-114)，pp. 31–36 (2008).

[18] 台坂　博，中里直人，石川　正，湯浅富久子：多倍長専用計算機 GRAPE9-MPX の拡張とその性能評価，情報処理学会研究報告. [ハイパフォーマンスコンピューティング]，Vol. 2015, No. 7, pp. 1–7（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110009877716/⟩ (2015).

[19] 台坂　博，中里直人，石川　正，湯浅富久子，似鳥啓吾：多倍長精度積分計算を加速させる専用システム GRAPE9-MPX の開発とその応用 (リコンフィギャラブルシステム)，電子情報通信学会技術研究報告 = IEICE technical report : 信学技報，Vol. 116, No. 417, pp. 13–18（オンライン），入手先 ⟨https://ci.nii.ac.jp/naid/40021100988/⟩ (2017).